Applying the Options Pattern

Steve Gordon

MICROSOFT DEVELOPER TECHNOLOGIES MVP

@stevejgordon www.stevejgordon.co.uk

Overview

Introducing the options pattern

- IOptions<T>
- IOptionsSnapshot<T>
- IOptionsMonitor<T>

Using named options

Options validation

Unit testing dependent types

Demo

Getting started with the options pattern

Binding configuration to strongly typed options classes

Injecting options with IOptions<T>

Demo

Reloading with IOptionsSnapshot<T>

Demo

Accessing options via IOptionsMonitor<T>

Demo

Using named options

- Defining named options
- Consuming named options

Demo

Options validation

- Data annotation attributes

When is validation applied?

Ensuring options are valid at startup

Demo

Options validation

- Defining a validation action
- Implementing IValidateOptions

Demo

Validating named options

Choosing Between the Options Interfaces

IOptions<T>

Does not support options reloading

Registered as a singleton in D.I. container

Values bound when first used

Can be injected into all service lifetimes

Does not support named options

IOptionsSnapshot<T>

Supports reloading of configuration

Registered as scoped in D.I. container

Values may reload per request

Can not be injected into singleton services

Supports named options

IOptionsMonitor<T>

Supports reloading of configuration

Registered as a singleton in D.I. container

Values are reloaded immediately

Can be injected into all service lifetimes

Supports named options

Choosing an Options Interface

	Use in singletons	Supports reloading	Named options
IOptions	✓	✗	✗
IOptionsSnapshot	✗	✓	✓
IOptionsMonitor	✓	✓	✓

Demo

Forwarding to options via an interface

Demo

Unit testing types dependent on options classes

- Using Options.Create
- Mocking with Moq
- Using IServiceProvider

Review

Options pattern

- Binding configuration to strongly typed options

Options consumption

- IOptions<T>
- IOptionsSnapshot<T>
- IOptionsMonitor<T>

Using named options

Applying options validation

Unit testing dependent types